Multi-agent active learning for distributed black-box optimization

Loris Cannelli, *Member, IEEE*, Mengjia Zhu, *Member, IEEE*, Francesco Farina, Alberto Bemporad, *Fellow, IEEE*, Dario Piga, *Senior Member, IEEE*

***Abstract*— Global optimization problems over a multi-agent network is addressed in the paper. The objective function, possibly subject to global constraints, is not analytically known, but can only be evaluated at any query point. It is assumed that the cost function to be minimized is the sum of local cost functions, each of which can be evaluated by the associated agent only. The proposed algorithm asks the agents at each iteration first to fit a *surrogate function* to local samples, and subsequently to minimize, in a cooperative fashion, an *acquisition function*, in order to generate new samples to query. In this paper we build the acquisition function as the sum of the local surrogates, in order to exploit the knowledge of these estimates, plus another term that drives the minimization procedure towards unexplored regions of the feasible space, where better values of the objective function might be present.**

The proposed scheme is a distributed version of the existing algorithm GLIS (GLobal optimization based on Inverse distance weighting and Surrogate radial basis functions), and share with it the same low-complexity and competitiveness, with respect to, for instance, Bayesian Optimization (BO). Experimental results on benchmark problems and on distributed calibration of Model Predictive Controllers (MPC) for autonomous driving applications demonstrate the effectiveness of the proposed method.

***Index Terms*— Black-box optimization, distributed optimization, Model Predictive Control, multi-agent networks, surrogate models**

I. Introduction

ACTIVE learning algorithms for black-box global optimization aim to optimize an expensive-to-evaluate objective function $f(x)$, whose analytical expression is typically not available and can only be evaluated through experiments or simulations [1]. Different active learning algorithms for black-box optimization with a minimum amount of function evaluations have been proposed in the last decades [2], [3], [4], and the most popular one is definitely *Bayesian Optimization* (BO) [1], which relies on successively constructing a probabilistic surrogate function (typically, a Gaussian Process) approximating the objective $f(x)$. The surrogate is then used at each iteration of BO to select the next point where to query $f(x)$, by trading-off exploitation (searching for values of x where $f(x)$ is expected to be optimal) and exploration (searching for points x for which $f(x)$ is highly uncertain). The same rationale is also adopted by other active-learning based approaches for black-box global optimization [2], [3], [4]. In this paper, we present a distributed version of GLIS, an active learning scheme for black-box optimization recently proposed in [3].

The need of distributed active learning algorithms for black-box optimization comes from the recent research interest in networked multi-agent systems [5], [6]. The commonality of network-structured applications is that they need to perform a decentralized optimization. This happens mainly because of two aspects: i) a lack of a central controller/authority, and ii) an inherent time-varying dynamics of the connectivity structure. Indeed, in several applications the presence of a central controller (a master node) is impractical or unattractive for various reasons: 1) its resources may be insufficient to coordinate the whole network and communicate with all the agents (e.g., limited bandwidth); 2) its single failure will cause the entire system to fail (robustness concerns); 3) privacy and confidentiality of the local information of the agents can not be preserved; 4) some of the agents in the network might be low-power devices that can communicate only with nodes in their physical proximity, making unfeasible the existence of a star topology (or a spanning tree). Furthermore, additional advantages of distributed systems is their inherent flexibility: the network topology and connectivity may be time-varying due, e.g., to agents mobility, link failures, or power outage.

The main idea of GLIS is to recursively build a surrogate of the objective function $f(x)$ as a linear combination of a *Radial Basis Function* (RBF), whose parameters are estimated by quadratic programming. The acquisition function used to select the query point x at each iteration of GLIS is then constructed as a weighted sum of the surrogate and of an *Inverse Distance Weighting* (IDW) function that is used to promote exploration of the input space. In the distributed version of GLIS presented here, called D-GLIS, we consider a black-box optimization problem in which N agents attempt optimizing a global separable objective given by the sum of local objectives $f_i(x)$, i.e., $f(x) = \sum_{i=1}^{N} f_i(x)$. Following the GLIS approach, at each iteration of D-GLIS, the i-th agent updates a surrogate $\hat{f}_i$ of its local objective f_i based on

This work was partly supported by HASLER STIFTUNG under the project *Black-box optimization in multi-agent networks: towards a distributed learning approach*.

Loris Cannelli and Dario Piga are with the IDSIA Dalle Molle Institute for Artificial Intelligence, SUPSI-USI, 6962, Lugano, Switzerland (e-mail:{loris.cannelli, dario.piga}@supsi.ch).

Mengjia Zhu and Alberto Bemporad are with IMT School for Advanced Studies, Lucca, Italy (e-mail: {mengjia.zhu, alberto.bemporad}@imtlucca.it).

Francesco Farina is an independent researcher with no affiliation (e-mail: farina@skiff.com).

its current estimate of the surrogate itself. Then, each agent optimizes a local acquisition function to find the next point to query locally. The local acquisition function consists of the combination of a surrogate of the global objective $f(x)$ and a local IDW function promoting the exploration of areas not yet explored by the i-th agent.

We refer to distributed optimization in terms of:

- experiments performed by the agents. In fact, the input space is cooperatively explored by the agents. Unlike other approaches for multi-agent learning [7], [8], we assume that the agents do not share information on their local objective f_i nor the corresponding surrogate $\hat{f}_i$;
- optimization of the local acquisition function minimized by each agent. Since each agent builds its own surrogate, and this information is not shared among the others, D-GLIS relies on decentralized consensus schemes. This allows the agents to cooperate over a network to achieve a global performance objective by exchanging a limited amount of local information with their one-hop neighbors only.

The two arguments above represent the main differences between GLIS and D-GLIS, and thus the main novelty of the paper. The rest of the paper is organized as follows. The active learning problem for distributed optimization is formulated in Section II. Section III presents the proposed distributed learning scheme D-GLIS. Section IV discusses the building blocks of D-GLIS that lie in the field of distributed optimization. Section V presents numerical results on benchmark global optimization problems (solved in a distributed fashion) and on decentralized calibration of an MPC for autonomous driving. The accompanying technical report [9] provides further details on the numerical distributed optimization algorithms used by D-GLIS, as well as more details on the numerical examples.

II. Problem formulation

Consider a network composed by N computing units (agents). We address the problem of solving the following optimization problem:

$$x^\star = \arg\min_{x\in\mathcal{X}} f(x), \tag{1}$$

where $x \in \mathbb{R}^n$ and $\mathcal{X} \subseteq \mathbb{R}^n$ is the set of feasibility which is supposed to be known. We assume that the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is separable:

$$f(x) \triangleq \sum_{i=1}^{N} f_i(x), \tag{2}$$

where $f_i : \mathbb{R}^n \to \mathbb{R}$ is the local cost function of the i-th agent. We suppose that:

- the analytical expression of the local functions f_i (with $i = 1,\dots,N$) are not available, and f_i can only be observed by the i-th agent through evaluation of $f_i(x)$ at any selected $x \in \mathcal{X}$, possibly in a noisy way;
- each agent performs its own function evaluation of $f_i(x)$, without sharing this information with the others. At the same time, the agents cooperate to meet the global objective of solving (1);
- evaluating $f_i(x)$ is expensive, and thus problem (1) needs to be solved within a limited number of function evaluations.

As an example, $f(x)$ can be a global performance index, which is given by the average of the local performance indexes $f_i(x)$ of each agent. The agents might not provide information on the local function evaluations $f_i(x)$ they performed for several reasons, such as for privacy issues, reduction of communication costs, lack of a central coordinator (master), etc.

The communication among the agents is modeled as a fixed, directed weighted graph $\mathcal{G} = (\{1,\dots,N\}, \mathcal{E}, \mathcal{A})$, where $\{1,\dots,N\}$ is the set of the vertices/agents, $\mathcal{E} \subseteq \{1,\dots,N\}\times\{1,\dots,N\}$ is the set of edges/communication links, and $\mathcal{A} \in \mathbb{R}^{N\times N}$ is the weighted adjacency matrix of the graph. The edge $(i,j) \in \mathcal{E}$ models the fact that agent i can send a message to agent j. $\mathcal{A}$ is compliant with the topology described by $\mathcal{E}$, that is to say, being α_{ij} the $(i,j)-$entry of $\mathcal{A}$, then $\alpha_{ij} > 0$ if $(i,j) \in \mathcal{E}$, and $\alpha_{ij} = 0$ otherwise. In the rest of this work it will be assumed that i) $\mathcal{G}$ is strongly-connected, and ii) $\mathcal{A}$ is doubly stochastic. These are two common minimal assumptions (see, for example, [10]) ensuring that the information of each agent influences the information of any other agent infinitely often in time.

In the following section we present D-GLIS, which aims at solving problem (1) through a decentralized strategy relying on the communication graph $\mathcal{G}$ and based on active learning.

III. D-GLIS

A. Local surrogate functions

Assume that each agent i has collected a local dataset $\mathcal{D}_i = \{x_j, y_j\}_{j=1}^{M_i}$ of length M_i, where y_j is the noisy observation of $f(x_j)$. The set $\mathcal{D}_i$ is not shared with the other agents. Among many possible choices to construct the surrogate $\hat{f}_i$ approximating the true unknown local objective f_i for each agent $i = 1,\dots,N$, we adopt the following weighted linear combination of RBFs, according to the original version of GLIS [11]:

$$\hat{f}_i(x) = \sum_{k=1}^{M_i} \beta_k^{(i)} \phi(\epsilon d(x, x_k)), \tag{3}$$

where $\phi : \mathbb{R} \to \mathbb{R}$ is an RBF, with $d(x, x_k)$ being any distance function between x and x_k and $\epsilon > 0$ a scalar hyper-parameter defining the shape of the RBF. The unknown coefficients $\beta^{(i)} = [\beta_1^{(i)}, \dots, \beta_{M_i}^{(i)}]^{\mathrm{T}}$ are determined by fitting the model $\hat{f}_i(x)$ to the dataset $\mathcal{D}_i$ through the minimization of the regularized squared error:

$$\sum_{k=1}^{M_i} \left\| y_k - \sum_{k=1}^{M_i} \beta_k^{(i)} \phi(\epsilon d(x, x_k)) \right\|^2 + \gamma \left\| \beta_k^{(i)} \right\|^2, \tag{4}$$

where the quadratic regularization term is added to guarantee strict convexity of (4). Some RBFs commonly used are $\phi(\epsilon d) = \frac{1}{1+(\epsilon d)^2}$ (*inverse quadratic*) and $\phi(\epsilon d) = e^{-(\epsilon d)^2}$ (*squared exponential kernel*), with hyper-parameter ϵ tuned through cross-validation.

It is interesting to note that minimizing the regularized cost in (4) bears resemblance to the distributed Support-Vector-Machine problem considered in the recent work [12]. However, unlike in [12], the surrogate functions $\hat{f}_i$ do not share the same parameters (i.e., $\beta_k^{(i)} \neq \beta_k^{(j)}$ for $i \neq j$), which means that the agents do not need to reach a consensus on the coefficients β_k.

Once local surrogate functions $\hat{f}_i$ are estimated, the surrogate $\hat{f}$ of the global objective f is simply:

$$\hat{f}(x) = \sum_{i=1}^{N} \hat{f}_i(x). \tag{5}$$

If the global surrogate $\hat{f}$ were known to all agents, it could be in principle minimized in order to find the new sample x_{T+1} at iteration $T+1$ of D-GLIS. However, two issues arise. First, we assume that the agents do not share their local surrogates $\hat{f}_i$. Thus, distributed algorithms must be adopted to minimize $\hat{f}$. This point will be discussed in Section IV. The second issue is due to the fact that, by considering only the surrogate $\hat{f}$, we only exploit the current available observations. Thus, the global minimum of problem (1) can be missed as the surrogate is not guaranteed to well approximate the true objective f in unexplored regions of the input domain $\mathcal{X}$. Therefore, a term promoting exploration of the input space $\mathcal{X}$ must be considered, which should be different for each agent, as the agents may explore the input space in different ways. This point is discussed in the following paragraph.

B. Local inverse distance weighting functions

According to [3], the IDW function is used to promote exploration. In particular, for the i-th agent, the IDW function is defined as

$$z_i(x) = \begin{cases} 0 & x \in \{x_1, \dots, x_{M_i}\} \\ \tan^{-1}\left(\frac{1}{\sum_{j=1}^{M_i} w_j(x)}\right) & \text{otherwise} \end{cases}$$

where $w_j(x) = \frac{1}{\|x - x_j\|^2}$ and $x_1, \dots, x_{M_i}$ are the datapoints contained in $\mathcal{D}_i$. Clearly, $z(x) = 0$ for all inputs already tested by the agent, $z(x) > 0$ in $\mathbb{R}^n \setminus \{x_1, \dots, x_{M_i}\}$, and z_i increases as the distances between x and the already tested inputs $\{x_1, \dots, x_{M_i}\}$ increases.

C. Local acquisition functions

Given an exploration hyper-parameter $\delta \geq 0$, the local *acquisition function* $a_i : \mathcal{X} \to \mathbb{R}$ is constructed in order to balance exploration and exploitation. Specifically, given the global surrogate $\hat{f}$ (unknown to agent i), define a_i as follows:

$$a_i(x) \triangleq \frac{\hat{f}(x)}{\Delta \hat{f}} - \delta z_i(x), \tag{6}$$

where $\Delta \hat{f}$ is the range of the surrogate $\hat{f}$, i.e., $\Delta \hat{f} \triangleq \max_{x \in \mathcal{X}} \hat{f}(x) - \min_{x \in \mathcal{X}} \hat{f}(x)$ and is used in (6) as a normalization factor to ease the selection of the exploration parameter $\delta \in (0, 1]$.

At each iteration of D-GLIS, one agent, say agent i, $i = 1, \dots, N$, is selected in a round-robin fashion and the input parameter $x^{i,*}$ to test for this agent is computed, according to GLIS, as the solution of the optimization problem:

$$x^{i,*} = \arg\min_{x \in \mathcal{X}} a_i(x), \tag{7}$$

where the superscript i in $x^{i,*}$ express the dependency of the query input with the i-th agent.
It is worth noticing that in constructing the acquisition function a_i in (6) the global surrogate $\hat{f}$ is considered by the i-th agent, while exploration is driven by a local IDW z_i. This is because the agents should cooperate to optimize the global objective f, while each agent should also evaluate its local objective f_i through its own exploration of the domain $\mathcal{X}$.

D. Iterative optimization

Once the new input $x^{i,*}$ is selected, i) the i-th agent evaluates $y_i = f_i(x^{i,*}) + \epsilon$, ii) the local surrogate $\hat{f}_i$ and IDW functions z_i are updated, iii) a new agent j is selected. This procedure is iterated until a maximum number of iterations $T_{\max}$ is reached. Finally, the optimal solution $x^\star$ is computed by only minimizing the final global objective $\hat{f}$, thus switching off the exploration term. The main steps of D-GLIS are summarized in Algorithm 1.

Algorithm 1 D-GLIS

Inputs: maximum number of function evaluations per agent $N_{\max}$; exploration parameter δ; constraint set $\mathcal{X}$; initial datasets $\mathcal{D}_i = \{x_j, y_i\}_{j=1}^{M_i}$, and initial surrogate functions $\hat{f}_i$ -constructed from $\mathcal{D}_i$- for all agents $i = 1, \dots, N$.

1: **repeat**
2: select agent i according to a cyclic rule
3: build the IDW function z_i in Section (III-B) from $\{x_j\}_{j=1}^{M_i}$
4: define the local acquisition function a_i in (6)
5: compute (7) via distributed optimization (see Section IV)
6: evaluate $y^{i,*} = f(x^{i,*}) + \epsilon$
7: update the local dataset $\mathcal{D}_i \leftarrow \mathcal{D}_i \cup \{x^{i,*}, y^{i,*}\}$
8: $M_i \leftarrow M_i + 1$
9: fit a new local surrogate $\hat{f}_i(x)$ based on $\mathcal{D}_i$
10: construct the global surrogate $\hat{f}(x)$ as in (5)
11: **until** maximum numbers of iterations $T_{\max} = NN_{\max}$ is reached
12: compute consensus $x^* = \arg\min_x \sum_{i=1}^{N} \hat{f}_i(x)$ via distributed optimization (see Section IV)

Output: consensus $x^\star$

Remark. It is also possible to consider a *parallel* and fully decentralized version of Algorithm 1 where all the agents compute at each iteration their acquisition functions, solve in a distributed way at the same time N optimization problems as (7), and finally update their datasets $\mathcal{D}_i$. In this case, there is no need to have a central controller enforcing the cyclic selection rule.

IV. Distributed optimization

In the D-GLIS algorithm described in the previous section, the generic agent i optimizes its own acquisition function a_i,

which also depends on the local surrogate functions $\hat{f}_j$ (with $j \neq i$) of the other agents. However, local surrogates of the other agents are assumed not to be known by the i-th agent. This requires to use distributed algorithms, where all the agents communicate with the others to optimize the local acquisition function a_i of the i-th agent.

In order to solve (7) in `Step 5` of Algorithm 1 in a cooperative fashion, the agents leverage the GTAdam [13] distributed algorithm. GTAdam is a distributed version of the popular Adam algorithm [14]. Adam is a gradient-like optimization scheme that solves problems in the form of (7) in a *centralized* way. At each iteration of Adam, a solution estimate x^k is updated by computing a descent direction which is properly adjusted using the gradient history. GTAdam [13] solves problems in the form of (7) over a network of agents by means of local computation and communication only, without any central coordinator. In order to make Adam distributed, the renowned *gradient tracking* algorithm [15] is encapsulated into the Adam framework. The strong connectivity of $\mathcal{G}$ ensures that the information of each agent can reach any other agent in the network, while the double stochasticity of $\mathcal{A}$ guarantees that the agents asymptotically will converge to the same stationary point (see [13] for details). However, because of the non-convex nature of problem (7), no guarantees of convergence can be stated.

Note that GTAdam in D-GLIS is implemented not only in `Step 5` for solving (7) as described above, but, similarly, in `Step 12` too. Details on numerical implementations of GTAdams for D-GLIS are provided in the accompanying report [9].

V. Examples

In this section we test the D-GLIS algorithm on benchmark optimization problems and distributed MPC design for autonomous driving. Python codes can be downloaded at `https://leon.idsia.ch/lib_download`. The *disropt library* [16] is used for the distributed operations.

A. Benchmark optimization problems

We test D-GLIS on four benchmark optimization problems, denoted as `brent`, `camelsixumps`, `hartman3` and `ls`. Problems `brent`, `camelsixumps`, and `hartman3` are defined in [17], while `ls` is a distributed least square problem. A full description of the problems is available in the extended version [9] of this work. As an example, we show here how to reformulate the `brent` problem to fit our distributed framework: number of agents $N = 3$; $x \in \mathbb{R}^2$; cost function:

$$f(x) = \underbrace{(x_1 + 10)^2}_{f_1(x)} + \underbrace{(x_2 + 10)^2}_{f_2(x)} + \underbrace{e^{-x_1^2 - x_2^2}}_{f_3(x)};$$

constraints $x_i \in [-10 \quad 10]$ $(i = 1, 2)$. The knowledge of the global optima is only used to evaluate the quality of the solution obtained by D-GLIS. In all the experiments the agents communicate over a fixed undirected graph $\mathcal{G}$, generated using an Erdős-Rényi random model (n, p) with $p = 0.3$ and n the number of unknown variables. The adjaceny matrix $\mathcal{A}$ of the graph is obtained through a Metropolis-Hastings weight model [18]. Algorithm 1 has been applied, running GTAdam as inner distributed solver in `Step 5` and `Step 12`.

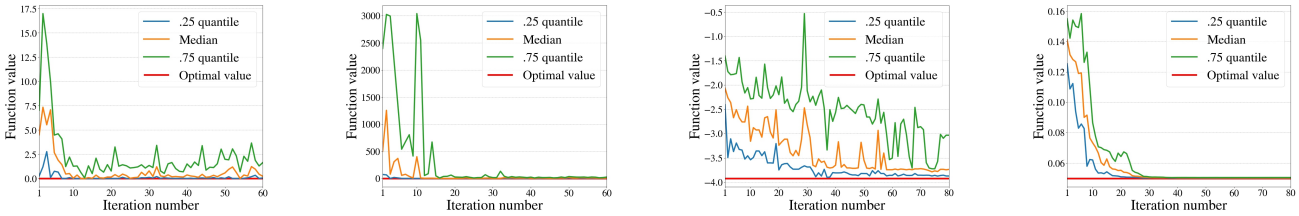
In all the experiments the agents start the optimization procedure with an initial local dataset $\mathcal{D}_i$ composed of $M_i = 2n$ points, $i = 1, \ldots, N$, generated uniformly at random in the feasible set. For each outer iteration of D-GLIS, the inner solver GTAdam runs for 1000 iterations. The value of the exploration hyper-parameter δ is updated by the agents while D-GLIS is running, according to the following heuristic: each agent i uses in `Step 5` a value $\delta_i \triangleq N \left(\max_{y_j \in \mathcal{D}_i} y_j - \min_{y_j \in \mathcal{D}_i} y_j \right)$, which depends on its current dataset $\mathcal{D}_i$. Each problem is run for 20 Monte-Carlo realizations and the performance of D-GLIS is shown in Figure 1 in terms of quantiles. The function values plotted in Figure 1 are obtained after each outer iteration of D-GLIS, by computing $x^* = \arg\min_x \sum_{i=1}^N \hat{f}_i(x)$ in a distributed way, and then evaluating $f(x)$ in $x^\star$. These points have been computed only for monitoring the performances of D-GLIS, but, in practice, $x^\star$ is computed only once, in `Step 12`, when the desired maximum number of iterations $T_{\max}$ is reached. Figure 1 shows that on all the tested problems D-GLIS approaches towards the global optimum after less than 80 experiments.

B. MPC for automated driving vehicles

As a case study, we use D-GLIS to calibrate an MPC for automated driving vehicles for lane-keeping and obstacle-avoidance. This case study was originally discussed in [19], [20] for MPC calibration through preference-based optimization. The test scenario is shown in Figure 2. A Subject Vehicle (SV) is on a one-way horizontal road with two lanes. Unlike the case study in [19], [20], we include two Obstacle Vehicles (OVs). Each OV is placed at the center of the lane and moves forward at a constant speed. OVs' initial velocities and initial longitudinal positions can vary, depending on the test scenarios. The SV is commanded by an MPC controller to follow the lane, and to avoid OVs by changing the lane, accelerating, or decelerating. Each calibrator (namely, agent) can prioritize optimization criteria differently and conduct various experiments. The goal is to reach consensus among them without disclosing the specifics of their experiments.

1) System description: A two-degree-of-freedom bicycle model, fully described in the supplementary material [9, Sec. 5], is implemented with state vector $s \triangleq [x_f, \ w_f, \ \theta]^{\mathrm{T}}$, where x_f and w_f (m) are the longitudinal and lateral positions of the SV's front wheel, and θ (rad) is the yaw angle. The manipulated variables $u \triangleq [v, \ \psi]^{\mathrm{T}}$, are the SV's velocity v (m/s) and steering angle ψ (rad), respectively.

2) MPC formulation: Full state observation is assumed and the control output y coincides with s. The prediction model of the MPC is a discretized version of the bycicle model, sampled with a sampling time $T_s = 0.085$. A linear time-varying MPC is designed via real-time iteration [21]. At each sampling time t, the following quadratic programming problem is solved



(a) `brent` (b) `camelsixhumps` (c) `hartman3` (d) `ls`

Fig. 1: Performances of D-GLIS on benchmark problems: function value vs. number of iterations.

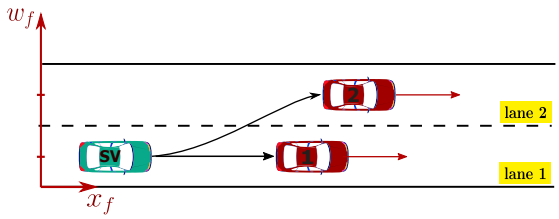


Fig. 2: Test scenario for MPC calibration.

according to receding horizon strategy:

$$\begin{aligned}\min_{\{u_{t+k|t}\}_{k=0}^{N_u-1}} : & \sum_{k=0}^{N_p-1} \left\|y_{t+k|t} - y_{t+k}^{\text{ref}}\right\|_{Q_y}^2 + \\ + & \sum_{k=0}^{N_p-1} \left\|u_{t+k|t} - u_{t+k}^{\text{ref}}\right\|_{Q_u}^2 + \sum_{k=0}^{N_p-1} \left\|\Delta u_{t+k|t}\right\|_{Q_{\Delta u}}^2, \\ \text{s.t.} \quad & y_{\min} \le y_{t+k|t} \le y_{\max}, \ k=1,\dots,N_p, \\ & u_{\min} \le u_{t+k|t} \le u_{\max}, \ k=1,\dots,N_p, \\ & \Delta u_{\min} \le \Delta u_{t+k|t} \le \Delta u_{\max}, \ k=1,\dots,N_p, \\ & u_{t+N_u+k|t} = u_{t+N_u|t}, \ k=1,\dots,N_p-N_u,\end{aligned} \tag{8}$$

where Q_y, Q_u, and $Q_{\Delta u}$ are weight matrices, $\Delta u_{t+k|t} \triangleq u_{t+k|t} - u_{t+k-1|t}$, y_{ref} and u_{ref} are, respectively, the reference values of control outputs and inputs during the experiments, and N_u and N_p are the control and prediction horizon.

3) Test scenarios and control objectives: We consider the following MPC parameters to calibrate: control and prediction horizons N_u and N_p; and the diagonal elements of the weight matrix $Q_{\Delta u} \triangleq \left[\begin{smallmatrix} q_{u11} & 0 \\ 0 & q_{u22} \end{smallmatrix}\right]$. These parameters are tuned within the intervals: $N_p \in [10 \ \ 30]$; N_u is taken as a fraction ϵ_c of N_p, with $\epsilon_c \in [0.1 \ \ 1]$; and $\log(q_{u11})$, $\log(q_{u22}) \in [-5 \ \ 3]$. The rest of the MPC parameters are fixed, with $Q_y = \left[\begin{smallmatrix} 0 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 1 \end{smallmatrix}\right]$, and $Q_u = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right]$. The reference value v^{ref} of the manipulated variable v is set to 50 km/h. During the experiments, v can fluctuate within the interval $[1 \ \ 90]$ km/h, with its rate of change $\dot{v} \in [-4 \ \ 4]$ m/s^2. The steering angle ψ can vary between -45° and 45° at a rate within $[-60 \ \ 60]$°/s, with its reference value $\psi^{\text{ref}} = 0°$. As for the control commands: $x_f \in [\text{-}\infty, \infty]$ m; $w_f \in [-0.6 \ \ 3.6]$ m to ensure that SV is within the road; and w_f^{ref} can take the two values 0 m or 3 m (namely, center of lane 1 or lane 2, respectively), depending on which lane the SV is on. The yaw angle θ is constrained to belong to the interval $[-90 \ \ 90]°$, and $\theta^{\text{ref}} = 0°$.

The following three objectives have been used to specify how the calibrated MPC controller should direct the SV to maintain lane position, prevent crashes with OVs and guarantee passengers' comfort: i) minimize the variation of the velocity; ii) minimize the variation of the steering angle; iii) avoid collision between SV and OVs. The expressions used to emulate the aforementioned objectives are: $f_{\text{m1}} \triangleq \frac{1}{N_{\text{total}}} \sum_{k=1}^{N_{\text{total}}} \left|\frac{v_k - v_k^{\text{ref}}}{v_k^{\text{ref}}}\right|$, $f_{\text{m2}} \triangleq \frac{1}{N_{\text{total}}} \sum_{k=1}^{N_{\text{total}}} \left|\frac{\psi_k - \psi_k^{\text{ref}}}{\psi_k^{\text{ref}} + 0.1}\right|$, and $f_{\text{m3}} \triangleq 1000 \ \mathcal{I}_{\text{collision}}$, where $N_{\text{total}} = 2\left\lceil \frac{t_{\text{exp}}}{2T_s} \right\rceil$ is the total number of discretization steps throughout the whole experiment, t_{exp} is the experiment duration, and $\mathcal{I}_{\text{collision}}$ denotes an indicator function which takes value 1 if SV and any OV collide, 0 otherwise.

We consider the presence of 4 calibrators (agents), each one weighting f_{m1} and f_{m2} with a different order of priority. The objective function f_i for agent i (with $i = 1, 2, 3, 4$) is defined below and normalized to the range of $[-2.5, 2.5]$:

$$\begin{aligned} f_1 &= 5(1 - \exp(-0.5 f_{\text{m1}} - 0.5 f_{\text{m2}} - f_{\text{m3}})) - 2.5, \\ f_2 &= 5(1 - \exp(-0.8 f_{\text{m1}} - 0.2 f_{\text{m2}} - f_{\text{m3}})) - 2.5, \\ f_3 &= 5(1 - \exp(-0.3 f_{\text{m1}} - 0.7 f_{\text{m2}} - f_{\text{m3}})) - 2.5, \\ f_4 &= 5(1 - \exp(-0.6 f_{\text{m1}} - 0.4 f_{\text{m2}} - f_{\text{m3}})) - 2.5. \end{aligned} \tag{9}$$

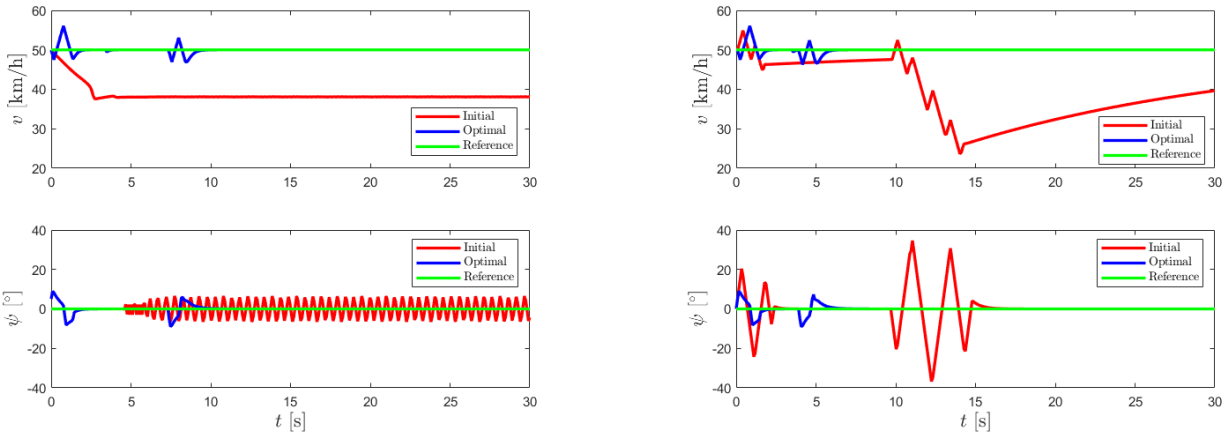
Furthermore, each agent assesses the MPC controller using different initial OV settings, i.e., different simulation experiments are performed by the agents. The calibration goal is to reach an agreement among them so that the MPC controller works well under diverse testing circumstances.

4) Calibration process: The calibration of the MPC parameters N_p, ϵ_c, $\log(q_{u11})$, and $\log(q_{u22})$ have been performed by running Algorithm 1 with configuration:

- the i-th agent only knows the local function $f_i(x)$;
- the agents communicate over a undirected graph $\mathcal{G}$, generated as in Section V-A;
- the initial local dataset $\mathcal{D}_i$ of the i-th agent is composed of $M_i = 2$ feasible points generated uniformly at random;
- GTAdam has been used as inner solver in `Step 5` and `12`.
- the value of δ is updated while D-GLIS is running, according to the same heuristic described in Section V-A;
- D-GLIS terminates after $T_{\max} = 80$ iterations.

For further clarifications, we remark that the QP problem (8) is solved locally by each agent, once the calibration parameters are fixed after each iteration of D-GLIS.

5) Results: D-GLIS provides MPC parameters with satisfactory performance with 16 learning experiments per agent. The optimized parameters of $[\epsilon_c, \ N_p, \ \log(q_{u11}), \ \log(q_{u22})]$ are $x^\star = [0.10, 20, \text{-}4.25, \text{-}5]^\top$. Figure 3 shows the



(a) `Test 1` (b) `Test 2`

Fig. 3: SV control performances obtained by the MPC in two test scenarios with random parameters (red curves) and optimal parameters $x^\star$ (blue curves): velocity v (top) and steering angle ψ (bottom) of the SV with respect to the simulation time t.

evolution of the velocity v and steering angle ψ obtained by the MPC with optimal parameters in two different test scenarios. For comparison, the velocity v and steering angle ψ achieved by an MPC with random hyper-parameters (extracted from the initial dataset $\mathcal{D}_3$) are also plotted. The MPC with random parameters generally exhibits more aggressive and frequent fluctuations in both manipulated variables, whose values also deviate from the reference. Other simulation scenarios are shown in the accompanying report [9].

VI. Conclusions

This paper has proposed D-GLIS, an algorithm to solve cooperatively, over a distributed network of agents, global optimization problems where the cost function is separable and expensive to evaluate, possibly subject to global constraints (known and inexpensive to evaluate). The proposed scheme, contrarily to many other approaches for black-box optimization, e.g., BO, is driven by deterministic arguments (the IDW function), which has the purpose of encouraging the investigation of unexplored regions of the feasible space. Differently from its predecessor GLIS, D-GLIS is distributed, meaning that a network of agents cooperates to solve a common optimization problem through distributed experiments, exchanging among themselves only fundamental information about the minimization procedure.

References

[1] E. Brochu, V. M. Cora, and N. de Freitas, "A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," *arXiv:1012.2599*, 2010.

[2] C. Malherbe and N. Vayatis, "Global optimization of Lipschitz functions," *International Conference on Machine Learning*, pp. 2314–2323, 2017.

[3] A. Bemporad, "Global optimization via inverse distance weighting and radial basis functions," *Comput. Optim. and Appl.*, vol. 77, pp. 571–595, 2020.

[4] L. Sabug Jr., F. Ruiz, and L. Fagiano, "SMGO: a set membership approach to data-driven global optimization," *Automatica*, vol. 133, pp. 109890, 2021.

[5] A. Nedić and J.s Liu, "Distributed optimization for control," *Annu. Rev. of Contr., Robot., and Auton. Sys.*, vol. 1, pp. 77–103, 2018.

[6] L. Cannelli, F. Facchinei, G. Scutari, and V. Kungurtsev, "Asynchronous optimization over graphs: Linear convergence under error bound conditions," *IEEE Trans. on Autom. Contr.*, vol. 66, no. 10, pp. 4604–4619, 2020.

[7] J. Choi, S. Oh, and R. Horowitz, "Distributed learning and cooperative control for multi-agent systems," *Automatica*, vol. 45, no. 12, pp. 2802–2814, 2009.

[8] J. Foerster, I. A. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," *Adv. Neural. Inf. Process. Syst.*, vol. 29, 2016.

[9] L. Cannelli, M. Zhu, F. Farina, A. Bemporad, and D. Piga, "Multi-agent active learning for distributed black-box optimization," 2023, Technical report TR-IDSIA-2023-02, `https://leon.idsia.ch/static/libraries/D-GLIS.pdf`.

[10] A. Nedić and A. Ozdaglar, "Distributed subgradient methods for multi-agent optimization," *IEEE Trans. on Autom. Contr.*, vol. 54, no. 1, pp. 48–61, 2009.

[11] A. Bemporad and D. Piga, "Global optimization based on active preference learning with radial basis functions," *Mach. Learn.*, vol. 110, pp. 417–448, 2021.

[12] M. Doostmohammadian, A. Aghasi, T. Charalambous, and U. A. Khan, "Distributed support vector machines over dynamic balanced directed networks," *IEEE Control Systems Letters*, vol. 6, pp. 758–763, 2022.

[13] G. Carnevale, F. Farina, I. Notarnicola, and G. Notarstefano, "Distributed online optimization via gradient tracking with adaptive momentum," *arXiv:2009.01745*, 2020.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980*, 2014.

[15] M. Zhu and S. Martínez, "Discrete-time dynamic average consensus," *Automatica*, vol. 46, no. 2, pp. 322–329, 2010.

[16] F. Farina, A. Camisa, A. Testa, I. Notarnicola, and G. Notarstefano, "Disropt: a python framework for distributed optimization," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 2666–2671, 2020.

[17] M. Jamil and X. Yang, "A literature survey of benchmark functions for global optimisation problems," *Int. J. Math. Model. Numer. Optim.*, vol. 4, no. 2, pp. 150–194, 2013.

[18] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," *Fourth Int. Symp. on Inf. Process. in Sens. Netw.*, pp. 63–70, 2005.

[19] M. Zhu, A. Bemporad, and D. Piga, "Preference-based MPC calibration," *Eur. Contr. Conf. (ECC)*, pp. 638–645, 2021.

[20] M. Zhu, D. Piga, and A. Bemporad, "C-GLISp: Preference-based global optimization under unknown constraints with applications to controller calibration," *IEEE Trans. on Contr. Syst. Technol.*, vol. 30, no. 5, pp. 2176–2187, 2022.

[21] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, "From linear to nonlinear MPC: bridging the gap via the real-time iteration," *Int. J. of Contr.*, vol. 93, no. 1, pp. 62–80, 2020.